A METHOD OF DATA STORAGE AND APPARATUS THEREFOR

BACKGROUND OF THE INVENTION

Field of the Invention

The present invention relates to the field of data processing, and more particularly, but not exclusively, to a method and apparatus for implementing a system which is able to provide digital storage services for public or corporate users.

Description of the Related Art

The explosive growth in automatic information systems and their interconnections via "cyberspace" has increased the dependence of both organizations and individuals on the information processed, stored and communicated using these systems. Some of the information, hereinafter called digital valuables or valuables for short - whether it be text, graphics, animation, video, audio, all types of data, or software (such as source code or machine code) written in various programming languages, are very precious for a user - whether the user is an organisation or individual, and need be kept in a safe place from which the valuables can be retrieved reliably and securely at a later stage. Examples of such digital valuables are patients' medical records, financial transactions, various certificates, proprietary business data, electronic money, and legal documents.

Currently, a user's digital information is commonly stored in the user's computer or in a networked file server. There are several fundamental problems with such an approach in that, firstly, it lacks integrity protection. Digital information is inevitably in digital format. The information is easy to tamper with, and the modifications need not leave any trace on the physical medium. In many situations it is necessary for a user to verify that the information retrieved is indeed what was stored in the first place and any modification to the information by system faults or malicious process can be detected. The second problem is low reliability. A hard disk or floppy disk may crash, resulting in irreversible loss of data. A networked file server provides higher reliability than a local hard disk but loss of

data is not uncommon. Moreover, such servers lack adequate security protection and are not easily accessible to public users. The third problem with the current technology of storing information is that it has little confidentiality protection.

It is an object of the invention to alleviate at least one disadvantage of the prior art.

SUMMARY OF THE INVENTION

According to the invention in a first aspect, there is provided a digital data depository for storing digital data items for a user comprising:
data storage means;
a user account associated with the user;
means for communication with the user;
means for authentication of the user with the depository;
means for establishing a digital data transaction session in which the user is able to instruct storage or retrieval of a digital data item in association with the user's account;
means for encoding the data item into a plurality of parts, the parts being separately stored in the storage means;and
means for decoding the encoded data item.

According to the invention in a second aspect, there is provided a method of storing digital data items for a user comprising the steps of:
providing a user account associated with the user;
authenticating the identity of the user;
receiving a digital data item and an instruction from the user for the item to be stored in association with the user's account; and
encoding the data item into a plurality of parts and storing the parts separately.

According to the invention in a third aspect, there is provided a method of protecting digital data comprising:

providing a data depository in which digital data may be stored electronically;

providing for registration of users of the data depository, each user having an

account with the depository;

in response to a request from a user, opening a transaction session with the user in which the user and the depository authenticate each other and performing a transaction instructed by the user in respect of a digital data item, the transaction being selected by the user from a plurality of available transactions including storage of the item in or retrieval of the item from the depository.

The described embodiment of the present invention discloses a method for implementing a digital valuables depository system, for public or corporate users to store and to retrieve precious digital information. The described embodiment is the electronic analogy to the physical safe boxes provided by banks whereby customers can keep their precious belongings. There are two generic entities in the system, a Service Provider (SP) and a User, the SP operating and provides Digital Safe services to the User. To make use of the service, the User first registers with the SP to open an account. The User can then deposit digital valuables into, retrieve and delete them from its account, all being carried out in an authentic and secure manner.

The SP ensures high reliability in storing users' digital valuables. To store a valuable, the SP first encodes each valuable into N parts based on an encoding algorithm, and then stores the N parts into one or more data storage devices. To retrieve or copy a valuable which has been stored previously, the SP reads the N parts from corresponding storage devices, and recovers the valuable from the N parts based on a decoding algorithm. The encoding and decoding algorithms are chosen such that the original valuable can be recovered correctly even if some of the N parts are lost or corrupted. To avoid storage error/corruption accumulation, the system periodically checks the N parts of every stored valuables and recovers/corrects lost/corrupted parts when they are detected.

BRIEF DESCRIPTION OF THE DRAWINGS

An embodiment of the invention will now be described, by way of example, with reference to the accompanying drawings in which:

FIG. 1 is a schematic diagram of a data storage system, being an embodiment of the invention.

FIG. 2 shows a possible data structure of the User Account maintained by the Service Provider (SP) in the preferred embodiment of the present invention.

FIG. 3 shows the steps to allow the User accessing his account in the preferred embodiment of the present invention.

FIG. 4(a) illustrates a possible logical structure of the Transaction Request message of type Valuable Storage (VS_Req) in accordance with the preferred embodiment of the present invention.

FIG. 4(b) illustrates a possible logical structure of the Transaction Request message of type Valuable Copy (VC_Req) in accordance with the preferred embodiment of the present invention.

FIG. 4(c) illustrates a possible logical structure of the Transaction Request message of type Valuable Deletion (VD_Req) in accordance with the preferred embodiment of the present invention.

FIG. 4(d) illustrates a possible logical structure of the Transaction Request message of type Valuable Retrieval (VR_Req) in accordance with the preferred embodiment of the present invention.

FIG. 4(e) illustrates a possible logical structure of the Transaction Request message of type Account Status Report (ASR_Req) in accordance with the preferred embodiment of the present invention.

FIG. 4(f) illustrates a possible logical structure of the Transaction Request message of type Session Close (SC_Req) in accordance with the preferred embodiment of the present invention.

FIG. 5(a) illustrates a possible logical structure of the Transaction Response message of type Valuable Storage (VS_Resp) in accordance with the preferred embodiment of the present invention.

FIG. 5(b) illustrates a possible logical structure of the Transaction Response message of type Valuable Copy (VC_Resp) in accordance with preferred embodiment of the present invention.

FIG. 5(c) illustrates a possible logical structure of the Transaction Response message of type Valuable Deletion (VD_Resp) in accordance with the preferred embodiment of the present invention.

FIG. 5(d) illustrates a possible logical structure of the Transaction Response message of type Valuable Retrieval (VR_Resp) in accordance with the preferred embodiment of the present invention.

FIG. 5(e) illustrates a possible logical structure of the Transaction Response message of type Account Status Report (ASR_Resp) in accordance with the preferred embodiment of the present invention.

FIG. 5(f) illustrates a possible logical structure of the Transaction Response message of type Session Close (SC_Resp) in accordance with the preferred embodiment of the present invention.

FIG. 5(g) illustrates a possible logical structure of the Transaction Response message of type Error (ERR_Resp) in accordance with the preferred embodiment of the present invention.

FIG. 6 shows the flow diagram of a Transaction Response Program (TRP) used in the preferred embodiment of the present invention.

FIG. 7 illustrates the flow diagram of the first Valuable Dispersal Program (VDP) used in

the preferred embodiment of the present invention.

FIG. 8 shows the flow diagram of the first Valuable Recovery Program (VRP) used in the preferred embodiment of the present invention.

FIG. 9 illustrates the flow diagram of the second Valuable Dispersal Program (VDP) used in the preferred embodiment of the present invention.

FIG. 10 shows the flow diagram of the second Valuable Recovery Program (VRP) used in the preferred embodiment of the present invention.

FIG. 11 shows the flow diagram of the first Valuable Checking and Correction Program (VCCP) used in the preferred embodiment of the present invention.

FIG. 12 shows the flow diagram of the second Valuable Checking and Correction Program (VCCP) used in the preferred embodiment of the present invention.

DETAILED DESCRIPTION OF THE INVENTION

Methods for implementing a digital data depository system,hereinafter referred to as Digital Safe, are described. The system is an electronic analogy to the physical safe boxes provided by banks whereby customers can keep their precious belongings. In the following description, numerous specific details are set forth such as logical structures of digital information and program steps, etc. in order to provide a thorough understanding of the present invention. It will be obvious to one skilled in the art that the present invention may be practised without these specific details. In other instances, well known steps as those involved in public key and symmetric key cryptosystem operations, in computing digest of a message using a one-way hash function, in digital signature generation and verification, in encoding of a data item into a codeword in an error-control coding scheme, in error-and-erasure-correction decoding, in erasure-correction decoding, are not shown in order not to obscure the present invention.

Notation and Nomenclature

The detailed description with respect to the implementation and operations of the Digital Safe system is presented partially in terms of algorithmic and symbolic representations of operations on data bits within the computer memory. These algorithmic descriptions and representations are the means used by those skilled in the art of data processing to most effectively convey the substance of their work to others skilled in the art.

An algorithm is here, and generally, conceived to be a self-consistent sequence of steps leading to a desired result. These steps are those require physical manipulation of physical quantities. Usually, though not necessarily, these quantities take the form of electrical or magnetic signals capable of being stored, transferred, combined, and otherwise manipulated. In this case, the physical quantities are voltage or current signals which correspond to the digital valuables/information being processed. It proves convenient at times, principally for reason of common usage, to refer to these signals as bits, values, elements, symbols, characters, terms, items, fields, numbers or the like. It should be borne in mind, however, that all of these and similar terms are to be associated with the appropriate physical quantities and are merely convenient labels applied to these quantities.

Furthermore, the manipulations performed are often referred to in terms such as adding or comparing, which are commonly associated with the mental operations performed by a human operator. No such capability of a human operator is necessary, or desirable. In most cases, in any of the operations described herein which form part of the present invention, the operations are machine operations. Useful machines for performing the operations of the present invention include general purpose digital computers or similar devices such as digital signal processors. In all cases, it should be borne in mind that there is a distinction between the method operation in operating a computer or other apparatus and the method of computation itself. The embodiment of the present invention to be described relates to method steps for secure and reliably storing users' digital valuables into and then subsequently retrieving them from a data depository system maintained and operated upon by a service provider.

The embodiment of the present invention also relates to an apparatus for performing these operations. This apparatus may be specially constructed for the required purpose or it may comprise general purpose computers as selectively activated or reconfigured by a computer program stored in the computers. The algorithms presented herein are not inherently related to any particular computer or other apparatus. In particular, various general purpose machines may be used with programs written in accordance with the teachings herein, or it may prove more convenient to construct specialized apparatus such as digital signal processor to perform the required method steps. The required structure for a variety of these machines would appear from the description given below.

GENERAL SYSTEM CONFIGURATION

A general model of the Digital Safe system is shown in FIG. 1. in which the Service Provider (SP) 10 provides data depository services to User 30 via a transmission channel 20. The SP system is responsible for storing and retrieving the User's digital valuables in a secure, reliable, and authenticated manner. A digital valuable, or valuable for short, has an unique identifier and is a self-contained entity. There can be various types of valuables including but not restricted in form to text, graphics, animation, video, audio, software, or any combination thereof. The transmission channel 20 represents the means and more specifically the media through which communication messages are exchanged between the SP 10 and the User 30. Such messages include the User's requests to the SP over paths 25 and 15 and the SP's responses to the User over paths 15 and 25. The transmission channel 20 includes but is not limited to any communication means or media such as computer networks, radio links, satellite links, diskettes or other storage medium. It should be understood by one skilled in the art that the term User is interchangeable with any user of information.

The described embodiment teaches methods for securely and reliably storing Users' valuables into and then retrieving them from the User's account maintained by the SP. These methods will be described with reference to specific steps of manipulating information. For one skilled in the art, it is obvious that some of these steps shall be best automated by, for example, implementing them as a special purpose software, which is

usually called a server, running on general purpose computers. It is clear that an information provider could simultaneously initiate multiple executions of the server to serve multiple end users. It is also clear that there may exist multiple SPs. For example, there may be one SP per organization or per district.

For clarity of presentation, the description below will elaborate on the model having one SP and one User. It is also clear that a User may also be another SP.

PREFERRED EMBODIMENT OF THE PRESENT INVENTION

1. Overall System Set-Up

Referring to FIG. 1, prior to using the Digital Safe services provided by the SP 10, the User 30 registers itself to the SP. During this registration process, the User authenticates itself to the SP by whatever means as required by the SP. The User agrees to the terms of a service contract. Such a contract contains at a minimum the identities, addresses of both the User and the SP, and the types of services to be provided by the SP. It may contain the public keys of the User and the SP, respectively. These keys are selected by the respective party based on certain public-key cryptosystems (PKCs). It may contain secret keys shared by the Users and the SP. The Digital Safe system relies on cryptographic protocols for mutual authentication between the User and the SP and for protecting the confidentiality and integrity of the User's valuables. Such cryptographic protocols require the User to possess cryptographic keys or secrets to operate. The contract may also specify the procedures for the User to recover lost cryptographic keys or secrets.

Message authentication code (MAC) is used to protect the integrity of a message. A MAC of a message can be generated using a symmetric key cryptosystem with the message and a secret value as inputs or using a one-way hash function with the message and a secret value as inputs. Since both digital signature and MAC are used for message integrity protection and authentication, we will refer them as Integrity Check (IC). For further references on PKC, symmetric key cryptosystem, generation and verification of digital

signature, one-way hash functions, generation and verification of MAC, and public key certificate, see D. E. R. Denning, Cryptography and Data Security, Addition-Wesley, Reading, MA, 1983; W. Stallings, Network and Internetworks Security - Principles and Practice, Prentice Hall, Englewood Cliffs, NJ, 1995; and C. Kaufman, R. Perlman and M. Speciner, Network Security - Private Communication in a Public World, PTR Prentice Hall, Englewood Cliffs, NJ, 1995.

At the end of the User registration, the SP sets up an account for the User. There are at least two types of accounts. In the first type, called flat-rate account, the User's account is allocated a fixed data storage quota and time interval (which may be extended or reduced at the User's request) and the service charge might be at a flat rate. In the second type, called flexible-rate account, the User's account has no fixed data storage quota and the User is charged by usage.

The User's account might be represented by a data structure shown FIG. 2, which includes a unique Account number (Acc_No) 50, the User's identity (U_ID) 52, contact information 54, optionally public keys or public key certificates 56, and information items 58 related to valuables stored in the account. Possible information items 58 are valuable identity (V_ID), type, size, time of submission, a protection flag (P_FLAG), an optional storage interval for flexible-rate account and pointers to locations where the N parts of the valuable are stored.

The P_FLAG takes two possible values, Encryption_Required and Encryption_Not_Required. If P_FLAG is set to Encryption_Required, then the valuable is encrypted by the SP and a pointer to the decryption key is kept in the account for each SP encrypted valuable. If P_FLAG is set to Encryption_Not_Required, then the valuable is not encrypted by the SP. The information items related to stored valuables will be empty initially and will be updated every time the User updates its account or when a valuable's storage interval expires. It should be noted that all the items in the User's account can be made visible to the User except the pointers to decryption keys and the pointers to storage locations, which are only accessible to and modifiable by the SP.

2. User Transaction Session with Service Provider

The Digital Safe services provided by the SP 10 to the User 30 include Valuable Storage (VS), Valuable Copy (VC), Valuable Retrieval (VR), Valuable Deletion (VD), and Account Status Report (ASR). The VS service lets the User store new valuables in its account; the VC service allows the User to get copies of its previously stored valuables from its account, the VD service lets the User remove one or more valuables from its account; the VR service permits the User to get copies of previously stored valuables and at the same time delete those valuables from his account; and the ASR service provides the User with its most recent account status report.

FIG. 3 shows the steps a user follows in accessing its account in the preferred embodiment of the present invention. In FIG. 3, the User and the SP starts a transaction session by running a mutual authentication and a session key exchange protocol. The User 10 prepares an Open Request (O_Req) message and sends the O_Req to the SP in step 100. The O_Req message is used to initiate the communications between the User and the SP, and more importantly, to authenticate the User to the SP. The O_Req contains at least the User's account number Acc_No 50 and/or identity U_ID 52. It is integrity protected by an IC (i. e., either the User's digital signature or a MAC) which can be verified by the SP. Upon receiving the O_Req, the SP verifies its validity in step 110. If it is not valid, an alert signal is sent to the User and the SP in step 120. If it is valid, the SP generates an Open Response (O_Resp) message, sends it to the User and opens a session with the User in step 130. The O_Resp is used to authenticate the SP to the User. This is integrity protected by an IC (i. e., either the SP's signature or a MAC) which can be verified by the User. After receiving O_Resp, the User checks its validity in step 140. An alert signal is sent to the User and/or the SP in step 150 if the Response is not valid; otherwise, the User proceeds to step 160. The objectives of O_Req and O_Resp are mutual authentication of the User and the SP, to negotiate a unique session identifier to bind the User's transaction with his account, and to negotiate session keys for encrypting and integrity protecting the follow on message exchanges during the entire session if the communication path between the User and the SP is not secure. There are many authentication and key distribution protocols in the literature (see, for example, C. Kaufman, R. Perlman, and M. Speciner, Network

Security - Private Communication in A Public World, PTR Prentice Hall, Englewood Cliffs, NJ. 1995) which fulfil these requirements. Such protocols can be used in place of O_Req and O_Resp.

Referring again to FIG. 3, following the successful opening of the transaction session, the User generates and sends a Transaction Request (T_Req) in step 160 to the SP. There are five types of T_Req messages corresponding to the five types of services: Valuable Storage Request (VS_Req), Valuable Copy Request (VC_Req), Valuable Deletion Request (VD_Req), Valuable Retrieval Request (VR_Req), Account Status Report Request (ASR_Req). In addition, there is a Session Close Request (SC_Req) which is used by the User to request closing of the current session.

The possible logical structures of the six T_Req types in accordance with the preferred embodiment of the present invention are given in FIG. 4(a) - 4(f).

FIG. 4(a) shows the structure of VS_Req. This logical structure comprises a session identifier (S_ID) 170 which was negotiated during steps 100 and 110 in FIG. 3 and is used here to identify uniquely the present session, a transaction type field VS_Req 175 which identifies the type of T_Req as Valuable Storage, a valuable identifier V_ID 180 which shows the identity of the submitted valuable, a field P_FLAG 182 which takes two possible values Encryption_Required and Encryption_Not_Required. The Encryption_Required value indicates that the User's valuable be encrypted by the SP before it is stored while the Encryption_Not_Required value indicates that the User's valuable be not encrypted by the SP before it is stored. The VS_Req message also contains a field V_By 185 which is the submitted valuable body, an optional field V_SI 188 which specifies the storage time interval of the submitted valuable (note that this field is not needed for flat-rate account), a freshness identifier F_ID_U 190 which guarantees that the message is fresh or in sequence, and a message IC field IC_U 195. The V_By 185 may be in plaintext or in ciphertext. The latter is preferred if the User wants to keep the valuable confidential only to itself. The V_By contains the User's digital signature which can be used by either the User, or the SP, or an independent judge to verify the authenticity of the valuable. To simplify the presentation, only one V_ID and one V_By are shown here. In general, the

message in FIG. 4(a) may contain multiple V_ID and multiple V_By fields.

FIG. 4(b) depicts the structure of VC_Req. It comprises the S_ID 170, a transaction type field VC_Req 205 which identifies the type of T_Req as Valuable Copy, a valuable identifier V_ID 210 which shows the identity of the valuable to be copied, a freshness identifier F_ID_U 215 and a message IC field IC_U 220.

Possible logical structures of VD_Req and VR_Req are given in FIG. 4(c) and FIG. 4(d), respectively. In FIG. 4(c), the field VD_Req 225 indicates the type of T_Req as Valuable Deletion and the field V_ID 230 shows the identity of the valuable to be deleted. In FIG. 4(d), the field VR_Req 245 indicates the type of the T_Req as Valuable Retrieval and the field V_ID 250 identifies the name of the valuable to be retrieved.

To simplify the presentation, only one V_ID field is shown in FIG. 4(b) - 4(d). In general, each message in FIG. 4(b) - 4(d) may contain multiple V_IDs.

FIG. 4(e) and FIG. 4(f) show the structures of ASR_Req and SC_Req, respectively. In FIG. 4(e), the field ASR_Req 265 indicates the type of T_Req as Account Status Report, and the field ASR_Option 270 specifies the format of the status report. Possible values of ASR_Option are FULL (in which case the SP will send the User a full version report, as specified in FIG. 2) and SHORT (in which case the SP will send the User a User selectable short version report). In FIG. 4(f), the SC_Req 285 indicates that the T_Req is of type Session Close.

In FIG. 4(a) to FIG. 4(f), the F_ID_U and the IC_U fields are optional. They should be present if the communication path between the User and the SP is not integrity protected. The F_ID_U may be a timestamp, a sequence number, or a nonce (a non-repeating random number generated by SP and sent to the User in advance). The IC_U is computed over the entire message; it may be the User's digital signature generated using its private key or it may be a MAC generated using a secret key which is shared between the User and the SP. The User's digital signature must be used for the IC_U field whenever non-repudiation of origin of the T_Req messages is a requirement.

Referring to FIG. 3 again, upon receiving the T_Req in step 490, the SP processes it and prepares a Transaction Response (T_Resp) message using a Transaction Response Program (TRP) in step 500. There are at least seven types of T_Resp messages: Valuable Storage Response (VS_Resp), Valuable Copy Response (VC_Resp), Valuable Deletion Response (VD_Resp), Valuable Retrieval Response (VR_Resp), Account Status Report Response (ASR_Resp), Session Close Response (SC_Resp), and Error Response (ERR_Resp). FIG. 5(a) - 5(g) show possible logical structures of the seven Transaction Response types in accordance with the preferred embodiment of the present invention.

FIG. 5(a) is the structure of VS_Resp. It comprises a session identifier (S_ID) 170 which was negotiated during steps 100 and 110 in FIG. 3 and is used here to uniquely identify the present session, a field VS_Resp 300 which indicates the type of T_Resp as Valuable Storage, V_ID 180 taken from FIG. 4(a), a field VS_ACK 305 which indicates the success or failure of the transaction processing, a freshness identifier F_ID_SP 310 which guarantees that the message is fresh or in sequence, and a message IC field IC_SP 315.

FIG. 5(b) is the structure of VC_Resp. It consists of S_ID 170, a field VC_Resp 320 which identifies the T_Resp as type Valuable Copy, V_ID 210 taken from FIG. 4(b), the copied valuable V_By 325 as specified by V_ID, a field VC_ACK 327 indicating the success or failure of the transaction processing, the fields F_ID_SP 330 and IC_SP 335.

FIG. 5(c) is the structure of VD_Resp. It consists of S_ID 170, a field VD_Resp 340 which identifies the T_Resp as type Valuable Deletion, V_ID 230 taken from FIG. 4(c), a field VD_ACK 342 indicating the success or failure of the transaction processing, the fields F_ID_SP 345 and IC_SP 350.

FIG. 5(d) is the structure of VR_Resp. It consists of S_ID 170, a field VR_Resp 355 which identifies the T_Resp as type Valuable Retrieval, V_ID 250 taken from FIG. 4(d), the retrieved valuable V_By 360 as specified by V_ID, a field VR_ACK 362 showing the success or failure of the transaction processing, the fields F_ID_SP 365 and IC_SP 370.

FIG. 5(e) is the structure of ARS_Resp. It comprises S_ID 170, a field ASR_Resp 375

which identifies the T_Resp as type Account Status Report, a field ASR_Report 380 with its format as specified by ASR_Option 270 in FIG. 4(e), a ASR_ACK field 382 indicating the success or failure of the transaction processing, the fields F_ID_SP 385 and IC_SP 390.

FIG. 5(f) is the structure of SC_Resp. It consists of S_ID 170, a field SC_Resp 395 which identifies the T_Resp as type Session Close, a field CS_ACK 398 indicating the success or failure of the transaction processing, the fields F_ID_SP 400 and IC_SP 405.

FIG. 5(g) is the structure of ERR_Resp. It consists of S_ID 170, a field ERR_Resp 410 which identifies the T_Resp as type ERROR, a field T_Req_Type 415, a field ERR_Status 420, the fields F_ID_SP 425 and IC_SP 430. T_Req_Type indicates the type of T_Req with which the ERROR message is associated with. Possible values of T_Req_Type are VS_Req, VC_Req, VD_Req, VR_Req, ASR_Req, and SC_Req. ERR_Status shows the type of errors which takes possible values such as "Request Not Verified" and "Request Not Permitted".

In FIG. 5(a) to FIG. 5(g), the F_ID_SP and the IC_SP fields are optional. They should be present if the communication path between the SP and the User is not integrity protected. The F_ID_SP may be a timestamp, a sequence number, or a nonce (i. e., a non-repeating random number generated by the User and sent to the SP before hand). The IC_SP is computed over the entire message; it may be the SP's digital signature generated using its private key or it may be a MAC generated using a secret key shared between the SP and the User. The former must be used if non-repudiation of origin of T_Resp messages are required.

Referring again to FIG. 3, the SP sends the T_Resp to the User in step 1000. The T_Resp sent in step 1000 by the SP is received at the User side in step 1020. The User first verifies if T_Resp is valid in 1030, including verifying the freshness of the freshness identifier F_ID_SP and the validity of the integrity check IC_SP. If the answer is "No", an alert signal is sent to the User and/or the SP in step 1040 and the User/SP will take actions accordingly. Assuming that the output of step 1030 is "Yes", the User then checks to see

if the received T_Resp is of type SC_Resp. If yes, the current session is closed; otherwise, it accepts and outputs the results of the T_Resp message in step 1055 and then goes back to step 160.

3. Operations of the Transaction Response Program

FIG. 6 shows the flow diagram of a Transaction Response Program (TRP) used in the preferred embodiment of the present invention. Referring to FIG. 6, the T_Req received in step 490 in FIG.3 is first checked for its validity in step 520, including checking the validities of the freshness identifier F_ID_U and the integrity check IC_U, as well as checking if the requested operation in T_Req is permitted. If the T_Req is not valid, a T_Resp of type ERR_Resp is formed in step 530, where the format of the ERR_Resp is as specified in FIG. 5(g). Assuming that T_Req passes the checking in step 520, it is next inspected in step 540 to see if it is of type SC_Req. If the answer is "Yes", the TRP closes the current session and prepares a T_Resp of type SC_Resp in step 550 according to the format of the T_Resp given in FIG. 5(f). Assuming that the outcome in step 540 is "No", the T_Req is checked in step 560 to see if it is of type VS_Req. If yes, the TRP first processes and stores the submitted valuable using a Valuable Dispersal Program (VDP), updates the User's account (i. e., records the valuable's V_ID, time of storage, storage duration, pointers to storage locations, the value of P_FLAG and pointer to the decryption key if P_FLAG = Encryption_Required), and then prepares a T_Resp of type VS_Resp in step 570 according to the format as specified in FIG. 5(a). If step 560 outputs "No", the T_Req is next inspected to see if it is of type VC_Rep in step 580. If yes, the TRP first recovers the requested valuable using a Valuable Recovery Program (VRP) and then prepares a T_Resp of type VC_Resp in 590, where the format of VC_Resp is according to FIG. 5(b). Assuming that step 580 yields "No", the T_Req is next checked in step 600 to see if it is of type VD_Req, if the answer is "Yes", the TRP first deletes the valuable specified in T_Req and then prepares a T_Resp of type VD_Resp in step 610 according to the format as given in FIG. 5(c). Assuming that the outcome of step 600 is "No", the T_Resp is next checked in step 620 for type VR_Req. If yes, the TRP first recovers the requested valuable using VRP, delete the valuable from the User's account, and prepares a T_Resp of type VR_Resp according to the format of FIG. 5(d), all being performed in

step 630. If the output of step 620 is "No", the T_Req must be of type ASR_Req. In this case, the TRP prepares a T_Req of type ASR_Req, with its format as given in FIG. 5(e). It should be noted that the T_Resp messages prepared in steps 530, 550, 570, 590, 610, 630, and 640 are sent to the User in step 1000 in FIG. 3. The executions of the steps 540, 560, 580, 600, 620, and 640 do not have to follow the order given above; they may be carried out in any order according to the preference if the system designer.

4. Operations of the First Valuable Dispersal Program and the First Valuable Recovery Program

FIG. 7 illustrates the flow diagram of the first Valuable Dispersal Program (VDP) used in the preferred embodiment of the present invention. The valuable to be processed by the VDP program has the valuable identifier V_ID 180 and valuable body V_By 185 as specified by the VS_Req message shown in FIG. 4(a). The valuable contains the User's digital signature which can be checked by the User and the SP to verify its authenticity. The basic function of VDP is to transfer a submitted valuable into N parts based on an (N, K) error-control code C with symbols over Galois field GF(2^m), where m is a positive integer. For further reference on encoding and decoding of a (N, K) error-control code, see R. E. Blahut, Theory and Practice of Error Control Codes, Addison-Wesley, Reading, MA, 1983. Also see S. Lin and D. J. Costello, Jr., Error Control Coding: Fundamentals and Applications, Prentice-Hall, Englewood Cliffs, NJ, 1983. Referring to FIG. 7, the User's digital signature on the valuable is first checked in step 700. If it is valid, the VDP proceeds to step 705; otherwise it goes to step 530 in Fig. 6. In step 705, the VDP checks if the P_FLAG 182 field of the VS_Req message is set to Encryption_Required. If "No", the program proceeds to step 715; otherwise, the valuable is encrypted under a cryptographic key in step 710. In step 715, the valuable or its ciphertext from step 710 is divided into q K-tuples, $X_i = (x_{i1}\ x_{i2}\ \ldots\ x_{iK})$ in step 715, for i = 1 to q, where xij is a symbol over GF(2^m). Each K-tuple X_i, i = 1 to q, is then encoded into a codeword $Y_i = (y_{i1}\ y_{i2}\ \ldots\ y_{iN})$ of the (N, K) error-control code C in step 720. Next, the q codewords Y_i, for i = 1 to q, are rearranged into N q-tuples, $Z_j = (y_{1j}\ y_{2j}\ \ldots\ y_{qj})$ in step 725, for j = 1 to N. Finally the q-tuples Z_j, j = 1 to N, are stored into one or multiple data storage devices.

FIG. 8 shows the flow diagram of the first Valuable Recovery Program (VRP) which works in conjunction with the first VDP in the preferred embodiment of the present invention. To recover the valuable (with identifier V_ID) which was dispersed by the first VDP in FIG. 7, the VRP first reads the N q-tuples $Z'_j = (y'_{1j}\ y'_{2j}\ \ldots\ y'_{qj})$, for j = 1 to N, from the corresponding storage devices in step 740 using the storage location pointers in the User's account, where Z'_j is the read version of Z_j. If Z'_j can not be found due to various reasons such as storage device crash or corruption, Z'_j is regarded as a q-tuple of "erasures", i. e., all the symbols in $Z'_j = (y'_{1j}\ y'_{2j}\ \ldots\ y'_{qj})$ are marked as "erasure". Next, the VRP rearranges N q-tuples Z'_j, j = 1 to N, into q N-tuples $Y'_i = (y'_{i1}\ y'_{i2}\ \ldots\ y'_{iN})$, for i = 1 to q, in step 745. It should be noted that if Z'_j is a q-tuple of "erasures", the jth symbol y'_{ij} in Y'_i is an "erasure". The N-tuple $Y'_i = (y'_{i1}\ y'_{i2}\ \ldots\ y'_{iN})$ is an erroneous codeword of the (N, K) code C, i.e., a codeword corrupted by errors and erasures. The q erroneous codewords $Y'_i = (y'_{i1}\ y'_{i2}\ \ldots\ y'_{iN})$, i = 1 to q, are input one by one to an error-and-erasure-correction decoder based on the (N, K) code C, to produce q K-tuples $X'_i = (x'_{i1}\ x'_{i2}\ \ldots\ x'_{iK})$, i = 1 to q, in step 750. If the errors and the erasures in $Y'_i = (y'_{i1}\ y'_{i2}\ \ldots\ y'_{iN})$ are correctable by the decoder, then $X'_i = X_i$. Step 755 checks to see if the q N-tuples are decoded successful. If not, an alarm is raised in step 765; if yes, the outputs of step 750, X'_i, i = 1 to q, are concatenated in step 760. In step 770, the VRP reads the corresponding P_FLAG from the User's account and checks if it equals Encryption_Required. If "No", the program produces the required valuable (which is the output of step 760) in step 780; if "Yes", the program proceeds to step 775 where it decrypts the output of step 760 using the appropriate decryption key. The result of step 775 is the required valuable which is output in step 780.

Let d denote the minimum Hamming distance of the (N, K) code C. Further, let e denote the number of erasures (i. e., lost symbols) and c the number of error symbols in $Y'_i = (y'_{i1}\ y'_{i2}\ \ldots\ y'_{iN})$, respectively. By the theory of error-control coding, the decoder output will be correct, i. e., $X'_i = X_i$, as long as $e + 2c < d$. In particular, for the so called maximum-distance codes, such as the Reed-Solomon codes, their minimum Hamming distance $d = N - K + 1$. Therefore, given K, an N can almost be selected, such that the probability that a valuable cannot be recovered is smaller than a pre-determined threshold.

5. Operations of the Second Valuable Dispersal Program and the Second Valuable

Recovery Program

FIG. 9 illustrates the flow diagram of the second Valuable Dispersal Program (VDP) used in the preferred embodiment of the present invention. The valuable to be processed by the VDP program has the valuable identifier V_ID 180 and valuable body V_By 185 as specified by the VS_Req message shown in FIG. 4(a). The valuable contains the User's digital signature which can be checked by the User and the SP to verify its authenticity. The basic function of VDP is to transfer a submitted valuable into N parts based on an (N, K) error-control code C with symbols over Galois field GF(2^m), where m is a positive integer. Referring to FIG. 9, the User's digital signature on the valuable is first checked in step 800. If it is valid, the VDP proceeds to step 810; otherwise it goes to step 530 in Fig. 6. In step 810, the VDP checks if the P_FLAG 182 of the VS_Req message is set to Encryption_Required. If "No", the program proceeds to step 815; otherwise, the valuable is encrypted under an encryption key in step 812. In step 815, the valuable (if P_FLAG = Encryption_Not_Required) or the output of step 812 (if P_FLAG = Encryption_Required) is divided into q K-tuples, $X_i = (x_{i1}\ x_{i2}\ \ldots\ x_{iK})$, for i = 1 to q, where xij is a symbol over GF(2^m). Each K-tuple X_i, i = 1 to q, is then encoded into a codeword $Y_i = (y_{i1}\ y_{i2}\ \ldots\ y_{iN})$ of the (N, K) error-control code C in step 820. The q codewords Y_i, for i = 1 to q, are rearranged into N q-tuples, $Z_j = (y_{1j}\ y_{2j}\ \ldots\ y_{qj})$ in step 825, for j = 1 to N. Next, an integrity check (as discussed under "Overall System Setup") IC_j is computed over Z_j, j = 1 to N, in step 830. Finally the (Z_j, IC_j), j = 1 to N, are stored into one or multiple data storage devices in step 835 with each Z_j being stored together with its corresponding IC_j.

FIG. 10 shows the flow diagram of the second Valuable Recovery Program (VRP) which works in conjunction with the second VDP in the preferred embodiment of the present invention. To recover the valuable (with identity D_ID) which was dispersed by the VDP of FIG. 9, the VRP first reads the N q-tuples $Z'_j = (y'_{1j}\ y'_{2j}\ \ldots\ y'_{qj})$ and the associated integrity checks IC'_j, for j = 1 to N, from the corresponding storage devices in step 840, where Z'_j is the read version of Z_j and IC'_j is the read version of IC_j. If Z'_j can not be found due to various reasons such as storage device crash or corruption, Z'_j is regarded as a q-tuple of "erasures", i. e., all the symbols in $Z'_j = (y'_{1j}\ y'_{2j}\ \ldots\ y'_{qj})$ are marked as "erasure". If Z'_j is found, it will be integrity checked using IC'_j. That is, an integrity check

is computed over Z'_j and the result is compared with IC'_j. If the two are equal, Z'_j is considered error free; otherwise, Z'_j is regarded as a q-tuple of "erasures". In step 845, the VRP checks to see if the number of erased Z'_js is less than d, the minimum Hamming distance of the (N, K) code. If not, it raises an alarm in step 850; if yes, the program proceeds to step 855 where the N q-tuples Z'_j, j = 1 to N, are rearranged into q N-tuples $Y'_i = (y'_{i1}\ y'_{i2} \ldots y'_{iN})$, for i = 1 to q. It should be noted that if Z'_j is a q-tuple of "erasures", the jth symbol y'_{ij} in Y'_i is an "erasure". The N-tuple $Y'_i = (y'_{i1}\ y'_{i2} \ldots y'_{iN})$ is a codeword corrupted by erasures. The q N-tuples $Y'_i = (y'_{i1}\ y'_{i2} \ldots y'_{iN})$, i = 1 to q, are input one by one to an erasure-correction decoder based on the (N, K) code C, to produce q K-tuples $X'_i = (x'_{i1}\ x'_{i2} \ldots x'_{iK})$, i = 1 to q, in step 860. As long as the number of erasures in Y'_i is less than d, the outputs of the decoder in step 860 will be correct. In step 865, the X'_i, i = 1 to q, are concatenated together. In step 870, the VRP reads the corresponding P_FLAG from the User's account and checks if it equals Encryption_Required. If "No", the program produces the required valuable (which is the output of step 865) in step 880; if "Yes", the program proceeds to step 875 where it decrypts the output of step 865 using the appropriate decryption key. The result of step 875 is the required valuable which is output in step 880.

6. Operations of Data Checking and Correction Programs

To avoid error accumulations in the N parts of the stored valuable, all stored data is checked periodically (in regular or irregular interests) by a Data Checking and Correction Program (DCCP). The DCCP checks if all the N parts of a valuable are intact. If not, it corrects the errors and recovers the missing parts.

There are two versions of DCCP. The first DCCP works in conjunction with the first VDP in section 4. Its flow diagram is shown in FIG. 11. In step 900, the DCCP collects the N q-tuples $Z'_j = (y'_{1j}\ y'_{2j} \ldots y'_{qj})$, j = 1 to N, associated with the valuable identified by a given D_ID. If Z'_j can not be found, DCCP marks Z'_j as a q-tuple of "erasures", j = 1 to N. It then rearranges Z'_j, j = 1 to N, into q N-tuples $Y'_i = (y'_{i1}\ y'_{i2} \ldots y'_{iN})$, i = 1 to q. In step 905, Y'_i is decoded into $X'_i = (x'_{i1}\ x'_{i2} \ldots x'_{iK})$, i = 1 to q, using an error-and-erasure-correction decoder of the (N, K) code C. Step 910 checks if all the q decoding operations in step 905 are successful. If not, an alarm is raised in step 915 and the program goes to

step 925; if yes, the DCCP proceeds to step 920. In step 920, X'_i is encoded into $Y''_i = (y''_{i1}\ y''_{i2} \ldots y''_{iN})$, i = 1 to q; the q N-tuples Y''_i, i = 1 to q, are rearranged into N q-tuples $Z''_j = (y''_{1j}\ y''_{2j} \ldots y''_{qj})$, j = 1 to N; finally, the old q-tuples $Z'_j = (y'_{1j}\ y'_{2j} \ldots y'_{qj})$ in the storage devices are replaced by the new ones $Z''_j = (y''_{1j}\ y''_{2j} \ldots y''_{qj})$, j = 1 to N. In step 925, the DCCP checks to see if there are more stored valuables need to be checked. If not, the program terminates; if yes, it gets the D_ID of the valuable to be checked in step 930 and goes back to step 900.

The second DCCP works in conjunction with the second VDP in section 5 and its flow diagram is shown in FIG. 12. For a given D_ID of a stored valuable, the DCCP collects the data segments (Z'_j, IC'_j), j = 1 to N, which are associated with the valuable in step 940. In step 945, the DCCP checks if all the N q-tuples $Z'_j = (y'_{1j}\ y'_{2j} \ldots y'_{qj})$ are found and if they all pass the integrity check based on IC'_j, j = 1 to N. If yes, the DCCP goes to step 975; if not, the DCCP proceeds to step 950 where the Z'_j is marked as a q-tuple of "erasures" if it is not found or if it fails to pass the integrity check. Step 955 checks to see if the number of erased Z'_js is less d, the minimum Hamming distance of the (N, K) code C. If not, the DCCP raises an alarm in step 960 and then goes to step 975; if yes, the program proceeds to step 965. In step 965, the Z'_j, j = 1 to N, are rearranged into $Y'_i = (y'_{i1}\ y'_{i2} \ldots y'_{iN})$, i = 1 to q, which are then decoded one by one using an erasure-correction decoder of the (N, K) code C, into $X'_i = (x'_{i1}\ x'_{i2} \ldots x'_{iK})$, i = 1 to q. In step 970, X'_i is encoded into codeword $Y''_i = (y''_{i1}\ y''_{i2} \ldots y''_{iN})$ of the (N, K) code C, i = 1 to q. The q codewords are then rearranged into q-tuples $Z''_j = (y''_{1j}\ y''_{2j} \ldots y''_{qj})$, j = 1 to N. Finally, the DCCP computes the integrity check IC''_j of Z''_j, and replace the old (Z'_j, IC'_j) with the new (Z''_j, IC''_j) in the storage devices, j = 1 to N. In step 975, the DCCP checks if there are any more valuables need to be checked. If not, the program terminates and if yes, it fetches D_ID of the valuable to be checked in step 980 and then goes back to step 945.

7. Procedures for Key Recovery

The User of the Digital Safe system needs to keep two sets of secret values, one for authentication to the SP in order to access its account and the other one for encrypting its valuables before submitting them to the SP. The former set of secret values are referred to

as User Authentication Secret Values (UASVs) and the latter set of secret values are referred to as Valuable Protection Secret Values (VPSVs). Examples of such secret values are private keys in public key cryptosystems and secret keys in symmetric key cryptosystems. All the secret values must be kept by the User in a secret and secure manner. An example of keeping a VPSV is to encrypt a valuable under the VPSV, encrypt the VPSV under a master key, and attach the encrypted VPSV with the valuable.

Loss of the UASVs prevents the User from accessing its account on-line. In the event of the User losing its UASVs, the User approaches pre-determined Key Escrow Agents to recover UASVs if UASVs are escrowed by such agents. Alternatively, the User will have to authenticate itself to the SP by other means (such as using paper documents) and upon authentication, the User and the SP separately or jointly establishing a new set of UASVs to replace the lost ones. The User from that point onwards uses the new UASVs to authenticate itself in order to access its account.

Loss of VPSVs prevents the User from decrypting its encrypted data or verifying the authenticity of the retrieved/copied valuables from its account. To prevent this from happening, the User should keep multiple copies of the VPSVs or make use of the key recovery services provided by a key escrow system. In the event of the User losing its VPSVs, the User approaches Key Escrow Agents to recover lost VPSVs. If the lost VPSVs are not escrowed, then valuables encrypted by the lost VPSVs will be lost too.